AIAA 2004-2166

Development of a Hybrid Model for Non-Equilibrium High-Energy Plasmas

J.-L. Cambier, M. Carroll
Air Force Research Laboratory, Edwards AFB, CA

M. Kapper
ERC, Edwards AFB, CA

35th AIAA Plasmadynamics and Lasers Conference
28 June – 1 July, 2004 / Portland, Oregon

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **07 JUN 2004**	2. REPORT TYPE	3. DATES COVERED **-**
4. TITLE AND SUBTITLE **Development of a Hybrid Model for Non-Equilibrium High-Energy Plasmas**		5a. CONTRACT NUMBER **F04611-99-C-0025**
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) **Jean-Luc Cambier; Marcus Carroll; M Kapper**		5d. PROJECT NUMBER **2304**
		5e. TASK NUMBER **0256**
		5f. WORK UNIT NUMBER **23040256**
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **ERC INCORPORATED,555 Sparkman Drive,Huntsville,AL,35816-0000**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR’S ACRONYM(S)
		11. SPONSOR/MONITOR’S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT
This paper summarizes the current effort at the Air Force Research Laboratory at Edwards AFB in developing a general hybrid model for the studies of multi-scale, non-equilibrium plasmas with high energy density. The software is being designed using Object-Oriented methods for maximal flexibility, portability and easy maintenance. The paper describes the general approach used in the model development, some of the problems to be solved, architecture and parallelization strategies, and some of the methods currently or being implemented. While still a work in progress, the software is rapidly building capabilities and will be applied in fundamental simulations of various plasma discharges.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**21**	

Development of a Hybrid Model for Non-Equilibrium High-Energy Plasmas

J.-L. Cambier[*], *M. Carroll*
AFRL/PRSA, Edwards AFB, CA 93524, USA
M. Kapper
ERC, Edwards AFB, CA 93524, USA

Abstract

This paper summarizes the current effort at the Air Force Research Laboratory at Edwards AFB in developing a general hybrid model for the studies of multi-scale, non-equilibrium plasmas with high energy density. The software is being designed using Object-Oriented methods for maximal flexibility, portability and easy maintenance. The paper describes the general approach used in the model development, some of the problems to be solved, architecture and parallelization strategies, and some of the methods currently or being implemented. While still a work in progress, the software is rapidly building capabilities and will be applied in fundamental simulations of various plasma discharges.

1. Introduction

The Air Force Research Laboratory has a significant interest in the development and evaluation of transient, high-power and micro-plasma discharges for compact high-performance electric propulsion systems, high-power conditioning, novel diagnostics methods, material fabrication, etc. The plasma conditions in these new generations of devices are usually extremely difficult to fully characterize experimentally or theoretically. Conventional numerical models are inadequate in those regimes, requiring the use of "hybrid models" that combine different numerical models for different components of the plasma, along with the interaction with electro-magnetic fields and materials. Hybrid modeling can be particularly challenging, as the model selection can be dynamically linked to the plasma conditions. Although there has been a considerable amount of research in developing advanced numerical methods and specialized plasma modeling tools for specific plasma problems, the lack of an *integrated* suite of software tools, combining various numerical approaches, is preventing the investigation of some types of problems. In a well-designed software environment, one should be able to use the same software tools and interfaces, irrespective of the plasma conditions being studied (e.g. grid generation, database, transport, visualization, diagnostics, etc), and be able to select the appropriate methods, or allow the software make the selection automatically. The current research effort is based on that concept, and is aimed at the modeling of plasma discharges characterized by: (a) extreme non-equilibrium conditions; (b) small spatial and temporal scales, and; (c) high-energy processes. In all the problems of interest here, the methods must include the interaction with local and applied electromagnetic (EM) fields and collisional momentum and energy exchange, as well as inelastic and radiative processes.

The general problem reduces to the evaluation of the velocity distribution function (DF) that describes the system and its dynamics, i.e. solving the Boltzmann equation:

$$\frac{\partial}{\partial t} f + \vec{v} \cdot \vec{\nabla}_x f + \vec{a} \cdot \vec{\nabla}_v f = CR(f, f') \qquad (1)$$

where $f(t, \vec{x}, \vec{v})$ is the velocity DF and the right-hand-side (RHS) symbolically describes all exchange terms due to collisional or radiative effects. However, a complete knowledge of the system DF is not possible given the current and even foreseeable computing capabilities, due to the hyper-dimensionality of the phase space, especially when plasma components with internal structure must be considered. A practical M&S capability for the multi-scale problems of interest here can only be achieved through hybrid modeling. In this approach, each plasma component is being treated by the numerical method that is the most appropriate and most efficient, given the overall physical conditions.

The plasma discharges of particular interest here are all characterized by a high-energy component far from equilibrium, but range from simple planar diodes to hollow cathode discharges, vacuum arcs, and intense laser-plasma interactions. The planar diode is an excellent verification & validation case, since it is extremely simple geometrically (1D problem), yet can lead to a rich spectrum of dynamical effects which can be investigated with analytical and other (Lagrangian) methods [1-2]. Another excellent test bed for hybrid models is the pseudospark discharge, which has been previously studied as a high-power plasma switch, and for which a similar attempt at analytical description of the ignition phase has been made [3].

We envision the integration of three non-exclusive categories of numerical approaches: (a) continuum representations, based on the solution of partial differential equations modeling a set of conservation laws; (b) particle methods, describing the plasma as a collection of individual "pseudo-particles", and; (c) direct kinetic equation solvers, which solve the dynamical equations (e.g. Vlasov, Fokker-Planck and Boltzmann equations) in a high-dimensionality phase space. The latter will be mostly used as a means of verifying the hybrid fluid-particle model. The particle models include both Particle-In-Cell (PIC) and Direct Simulation Monte-Carlo (DSMC). The PIC model [4] is initially electrostatic, non-relativistic only, and will be gradually upgraded to a fully relativistic, electro-magnetic PIC. The coupling between fluid and stochastic solvers proceeds through collisional exchanges of mass (e.g. ionization), momentum and energy; these are treated through a Monte-Carlo Collisions (MCC) algorithm, and variations of the basic procedure are being investigated. The DSMC [5] model will be added later. The fluid models currently being implemented span a range of non-equilibrium descriptions, starting from a single-fluid and single-temperature (thermal equilibrium) model all the way to multi-fluid and Collisional-Radiative (C-R) models.

The paper is organized as follows. In section 2, the approach to the software development is described, from general considerations to numerical and data representation issues, language, platform, parallelization and performance. Section 3 describes some of the models and algorithms implemented to date, as well as some integration issues. Section 4 describes planned future work and summarizes the development status.

2. Approach

2.1. General Development Strategy

In most cases of interest here, the plasma can be decomposed into a "bulk" plasma and a highly energetic component (electrons *and/or* ions). If the bulk plasma is strongly collisional (i.e. sufficiently dense), it can be modeled as a fluid, albeit at various levels of thermal non-equilibrium. The energetic component can itself be a function of time and space; for example, at the onset of the discharge there can be very few energetic particles; during a mid-phase the energetic component is maximum, until it finally relaxes through collisional exchange with the bulk and disappears. One could also envision a reverse process, where energetic particles are generated from the bulk (e.g. stochastic heating) itself. Short of modeling the entire DF in phase space, one would need to devise methods for transferring information back and forth between the bulk and energetic components. This is the general problem of mass, momentum and energy coupling between the dual representations (fluid and particles).

This issue is exemplified in previous attempts at pseudospark modeling with a hybrid method [6]; in that model, an energy threshold is assumed (the ionization potential) such that the plasma is modeled as a fluid below it, and as particles above it. However, there are some potential problems with this approach. First, there is no definite energy threshold of behavior: if the threshold is the ionization potential, this implies that all electronic excitations are being ignored. If it includes the minimal gap for electronic excitation, this does not include excitation of other modes for molecular plasmas. Furthermore, that particular model did not treat new generations of electrons (liberated by ionizing collisions) as particles, and while the ballistic approximation is efficient in separating the time scales of the electrons and ions, it does not allow for immediate field-coupling effects on the high energy electrons, which might be necessary for the beam phase. It is clear that a better hybrid model may be necessary in order to successfully model all phases of the pseudospark operation, as well as other high-energy and/or micro-discharges, and this may require that a consistent mechanism for particle creation and absorption from and into the bulk be devised.

Another problem to address concerns the collisional models. One can model collisions between the energetic particles and the background gas through a Monte Carlo Collisions (MCC) procedure. For electron collisions, the procedure is similar to a test-particle Monte-Carlo [7]: due to the difference in masses, the target atom can be assumed unperturbed during the collision. In the case where the energetic particle is an ion itself, particles must be sampled from the equilibrium bulk distribution, collided with the energetic particles, and the momentum and energy exchanged monitored; this approach does exactly conserve momentum and energy, but is expensive. Another approach is to use the more efficient “collision-field” method [8-10], which can also conserve momentum and energy but averages the velocity dependence into a collision frequency; this approach can also work for electron collisions. Finally, there is the interesting alternative of extended pseudo-particles, each one carrying its own internal distribution at an individual “temperature” [11]. The standard MCC is currently being implemented; at a later date, we plan to also implement the collision-field approach and compare the results.

It is also important to be able to validate the hybridization and collisional methods implemented in the proposed code; for that, we will also develop and implement some simple Vlasov and Fokker-Planck kinetic solvers with high resolution in phase space. This will allow us to test procedures for the exchange of particle and bulk components (without collisions). Furthermore, we will also implement a multi-fluid solver, which will model the electrons, ions and neutrals separately as fluids, with friction and electromagnetic coupling. This will allow us to test the limits at which the fluid model starts to break down.

Given the disparity of the modeling approaches and the complexity of the physics, it does not appear possible that all approaches can be combined into a single code. Such a code would be extremely large; very difficult to use even on simple problems; extremely difficult to maintain and upgrade, and using massive amounts of computational resources. It is much more appropriate to develop a *library* of modeling tools, specialized for plasma simulations, along with the software required for the modeling of the specific applications of interest. Therefore, other researchers can import these tools or integrate them to construct another code designed for another set of applications. In addition, the software should have a number of features aimed at maximizing its usefulness, such as: multi-dimensional capability (1D, 2D, 3D); ability to handle complex shapes and boundary conditions; simplified maintenance and upgrade; portability; high computational efficiency, and; graphical user interface (GUI) and visualization tools. These objectives are a part of the overall software development strategy.

2.2. Software Architecture and Parallelization Strategy

To have a significant impact on problems of practical interest, one must be able to efficiently combine the most important types of numerical approaches used for plasma modeling. Therefore, one must construct the data-structures and associated methods to be used in the software framework: these include geometry, flow, field variables, and particles. Generalized classes of solvers, easily re-configurable, must be constructed, and the coupling physics and interface procedures must be implemented. For a number of reasons, Java has been chosen as the language for the design and development of the software architecture. Java is a strong Object Oriented (OO) design language with an easy syntax that allows us to easily develop, debug and share software modules. Portable visualization and GUI can be completely integrated using open-source software, and there are no issues with compiler compatibilities between different vendors and platforms. Furthermore, Java allows us to easily test various parallelization approaches across different platforms, in a way that is compatible with the modern approach of “grid-computing”. Using a combination of Remote Method Invocation (RMI) and multi-threading, one can examine both distributed and shared memory parallelization, and have remote steering and real-time visualization. Except for a few exceptions [12-14], Java has so far not been considered as a viable alternative for large-scale scientific computing because of the perceived lack of performance. However, recent versions of the Java Virtual Machine (JVM) have come a long way to improve this situation; our own tests, confirmed by third-party developers, indicate that at least on some platforms (Intel) the execution speed for simple floating-point calculations are similar to optimized C. More complete performance comparisons have been performed elsewhere with similar conclusions [15]. Differences of even 10-30% in execution speed may not be a critical factor when compared to other issues such as platform availability and cost, whether hardware (the ability to use existing networks of disparate workstations has a significant cost advantage, especially given the rapid obsolescence of computer technology) and personnel costs for development and maintenance. A later version of the software may be written in C++ if it turns out to be beneficial – the translation from Java to C++ is not expected to present major difficulties. It is also expected that the simulations will be computationally demanding (especially in 3D); therefore, it is necessary to implement parallelization strategies as early as possible (once the data-structures and code architecture have been stabilized), while maintaining a high degree of flexibility.

There are several levels of parallelization that could be applied to the code. The current trends in computing and communication technologies increasingly favor the development of clusters of computing nodes, each node having an increasing number of shared-memory, hyper-threaded processors. Such clusters are routinely available with up to 128 nodes of 2 CPU each, with specialized machines having thousands of nodes. Multi-processor workstations are also readily available platforms that can be tapped for low-cost computing. It is therefore likely that an efficient parallelization strategy should plan for a re-configurable network of multiple platforms and a hierarchy of distributed and shared-memory architectures; this has been a major factor in choosing Java as the top-level language for the software architecture. Drawing from the "grid-computing" paradigm, we have created a client-master-server architecture (see Figure 2) such that computations can be started on any available platforms, and monitored in real-time by the user/client. This is similar to the JavaGrid approach described in [12], albeit at a lower level of complexity and capability.

To keep maximum flexibility in mind, the code is designed to make inter-node communication as transparent as possible. This is accomplished by the implementation of Java's remote method invocation (RMI) for all inter-processor communications [16]. Java-RMI allows for the manipulation of remote objects on a local machine. As an example, a client can invoke remote methods on the master, asking the remote method to return an object to the client, such as the computation data for visualization. The RMI architecture of the code consists of three levels—client, master, and server. The code is initialized on the master with a list of server(s) that will handle the computations. A "server" is a single-term denomination that can physically be, for example, a single-processor or multi-processor workstation, or a node in a Linux cluster[†]. Once the master has been initialized, the client can establish a connection to it [‡]. The client can then perform preprocessing operations such as set-up of the grid, boundary conditions, chemistry, etc, to initialize the computations and engage in real-time visualization of the data as it is being computed.

The software of course supports configurations from single-processor workstations to multi-processor shared-memory workstations as well as multi-processor distributed-memory clusters, in compliance with the grid-computing paradigm. This is made possible by the multi-threading capability of the server code, which allows for multiple blocks (each block being a subset of the computational domain) to be computed on each "server". Each block runs in its own thread, and as a result, multiple blocks on a single processor are computed in parallel. Multiple servers can be initialized with multiple blocks residing on each server. Only one server instance is needed per computer node, and as many servers can be initialized as there are blocks. The computation iterations are carried out on each server along with inter-server communications.

The synchronization of all iterations and communications is the responsibility of the master. The master utilizes various locks and the associated *wait/notify* methods inherited from `java.lang.Object` to determine when all of the servers have finished their computation and communication iterations and to cue them so that they may proceed with the next iteration. Once all of the blocks have been locked, the master is notified, and in turn, notifies all the blocks to begin the next iteration. The master is also in charge of global data which is processed from local data from each block, such as the time step, or data that must be visualized. The client code provides a graphical user interface from which the code can be controlled and data can be viewed in real-time. The client can make requests to the master to grab data from the servers as it is being computed and display it in real-time with VisAD—a Java visualization API built upon Java3D [17]. It can be selected from the client as to which blocks to view data from so that the communication overhead does not bog down the overall computation time.

The most crucial concern regarding communication overhead is the optimization of the intra-/inter-server communications. At every time-step, it is necessary for each block in the domain to share its information with one or more neighboring blocks. If two neighboring blocks are being computed on the same server, the communication can be handled by standard local method calls. However, if two neighboring blocks are computed on different servers, then inter-server communication is involved and it becomes necessary to use Java-RMI or other suitable message passing protocol. Figure 3 illustrates how two neighboring blocks communicate with each other. There are two types of patches which allow data transfer between two

[†] One should think of the "server" as a computational unit that provides a "service" (computations). There is a-priori no reason why the "server" cannot by itself further distribute the computational load to other service providers, and although this recurrent (multi-level) parallelization is not yet implemented, this is the direction taken here.

[‡] The master is a command and control center, responsible for gathering and distributing information to the servers and client, scheduling the tasks, providing critical I/O, and monitoring commands from the client.

blocks. The first of these being the source patch is an array of objects which contain references to data available in the boundary layer of cells. The data references are depicted by the dashed arrows in Figure 3. The second is the target patch from which the ghost layer of cells receives its information. During block communication, there is a method call for each source/target patch pair in which the target patch is filled with the information from its neighboring block's source patch. The method calls are depicted by the solid arrows which can either be a local call if both blocks are on the same server or an RMI call if each block is on a separate server. The exchange of boundary condition information between servers is done without involving the master: this prevents a bottle-neck when scaling to a large number of processors.

2.3. Geometry Data-Structure

The first compatibility issue one must deal with concerns the grid structure. Ideally, the software should be grid-independent; this implies that one could at first decide to implement solvers based on structured grids, while reserving the ability to implement unstructured grids and their solvers at a later stage. All numerical approaches described above make use of a discretized space: besides the self-evident case of Computational Fluid Dynamics (CFD), PIC is a typical particle-mesh method, and DSMC needs a mesh to determine collision partners. Grids can be described with the following categories: (1) Structured (constant number of grid points in each direction), or (2) Unstructured (triangles, tetrahedra). The former also has the following sub-cases: (1-a) Cartesian (Δx, Δy, Δz are constants); (1-b) Uniform (spacings are not constants); (1-c) Curvi-linear, or "body-fitted" (not necessarily orthogonal). Structured grids can be described by multi-dimensional arrays of given dimensions (`Q[][][]`), i.e. any grid point can be addressed by a set of indices (*i,j,k* in 3D). In unstructured grids there is no such ordering, and the grid connectivity (i.e. how to access another point from the current one) requires a list of pointers between grid elements. Despite the apparent difference between the two descriptions, one can use the same approach in representing the grid data and connectivity. Thus, one can define an object *gNode* to represent a grid point (a node in the mesh); each *gNode* contains the coordinates of the point (in real or transformed space), as well as a list of pointers (addresses in the memory where the data is located, or integer indices that indicate a position in an array) to the neighboring points. The same goes for object of type "*gCell*", which describes the volume enclosed by the grid lines or grid surfaces. Thus one can walk through the connected list of nodes and cells to reach every point, whether the grid is structured or not. Of course, the structured case has the additional advantage that one could also directly address a given node or cell from the set of indices *(i,j,k)*. However, instead of forcing the data to be stored in a structured form, it is just as easy to have a method that provides the memory location of a node/cell with those particular indices. Thus, the data structure (a one-dimensional list of objects) can remain the same to describe both structured and unstructured grids, while the additional methods "*getNode*" and "*getCell*" allow a mapping between the structured indices (*i,j,k*), arguments to the method, and the list. Such a mapping is of course void in the unstructured case.

Within this approach a "*Mesh*" object is defined as a list (or 1D array) of cells, nodes and surfaces, with pointers to connected objects. If a structured solver is being used, one must address information on either side of a cell or surface by the pointers to the corresponding objects, e.g. in 2D, `U[i-1][j]` is obtained from `U[ncL]`, with `ncL=pointer_left_cell(nc)`. This procedure has two disadvantages with respect to a simple multi-dimensional data structure (e.g. `U[][]`): (a) there is redundant information being stored in the *Mesh* object; however, the information is contained in a few variables and does not constitute any significant penalty; (b) there is additional overhead in, for example, the method call "*getCell*", which maps the one-dimensional index of the list of cells with the structured indices *(i,j,k)*; however, one can easily construct algorithms such that this method call is relatively infrequent and the cost is minimal[†] compared to the large number of operations involved in a flow or field solver, for example. The major advantage is that this approach allows the same data-structure to be used for any type of grid structure. The *Mesh* object can be sufficiently general that each cell has an arbitrary connectivity, while some variables used only for some types of grids may remain un-instantiated for others. For example, a structured grid may need a metric for some solvers; this metric may be computed and stored in that case, but is empty when the grid is unstructured. Similarly, some techniques of dynamic grid adaptation may be valid only for some types of grids (e.g. octree subdivision for Cartesian grids) and not on others. Each Mesh object is therefore flagged by a few Boolean variables (e.g. Cartesian? Curvilinear? Unstructured?), so that one can match appropriate solvers and methods to the types of grids being used. Even if not all solvers are immediately

[†] In some languages, in-lining can be used to eliminate the overhead associated with a function call.

implemented, future extensions are facilitated because the geometry infrastructure can support it. This is the approach used here: while the current focus is on developing solvers for structured, body-fitted grids, unstructured grid implementations will be facilitated in the future, using solvers and grid generators (which must provide the connectivity information) from external sources.

Structured, body-fitted grids are efficient for CFD, especially when boundary layers are present; in that case, cells can be stretched to match the preferred direction of the gradients (normal to the body surface), while allowing efficient tri-diagonal implicit solvers and increasing the accuracy of dimensionally-split high-order schemes. This approach may also be efficient for plasma sheaths, since again the thermal, plasma composition and plasma potential gradients are aligned with the normal to the body surface. On the other hand, Cartesian grids are preferable for particle transport, the construction of simple high-order schemes (e.g. for the Maxwell equations), Fourier-transformed schemes, and dynamic cell adaptation. If the grid is Cartesian, it is extremely easy to keep track of the cell to which the particle belongs, given its position, e.g: `i=x/Δx`. This can be important for a large number of non-interacting particles, where transport (particle "pusher") is the major computational activity. However, this is true mostly for collision-less plasmas in simple geometries[†]. When the plasma is collisional, most of the computational cost is in the MCC procedure. In those cases, the additional cost in determining the relative particle position with respect to the cells (i.e. cell-indexing the particle) for non-Cartesian geometries becomes a negligible fraction. This extra-cost is typically a few arithmetic operations, and is roughly of the same order for body-fitted grids (using metric-transformed variables) and unstructured grids (computing relative distances), as long as the particle moves at most to neighboring cells. This last limitation is also not a restriction for collisional plasmas, since it is important for solution accuracy. One should point-out that the two approaches can be effectively combined, by using body-fitted grids near the body and in collisional regions, and Cartesian grids away from the body and in collisionless regions. The ability to handle any grid type, as described above, is essential to this optimization scenario.

2.4. Fluid Data-Structure

The data-structure for the fluid and plasma variables must also be highly flexible. The fluid can be described at various levels of non-equilibrium, corresponding to various relaxation time-scales. By assumption, the velocity distribution of the fluid-like plasma components remains Maxwellian, the collisional momentum exchange between particles of similar masses being the fastest relaxation process. However, electrons and ions may have different velocity distributions (i.e. different translational temperatures), and internal modes may not follow identical equilibrium distributions, or may significantly deviate from Boltzmann distributions. This hierarchy of non-equilibrium descriptions results in a variety of multi-temperature models. The hierarchy for atomic plasmas would be single-temperature, two-temperature ($T_h - T_e$), three-temperature ($T_h - T_e - T_x$), and Collisional-Radiative (C-R) models, where T_h is the translational temperature of heavy particles, T_e is the temperature of the free electrons, and T_x is the temperature of the (assumed) Boltzmann distribution of electronic excited states. In the two-temperature model, the bound electronic states are assumed to be in a Boltzmann distribution at the temperature T_e; this is usually valid in cases where the electron collisions dominate over atomic collisions. In low-density cases, when radiative rates are comparable to collisional or atomic quenching becomes significant, the electronic states cannot be described by a Boltzmann distribution; in this regime the C-R model is required.

The ability to generate a variety of non-equilibrium descriptions is another important aspect of the code flexibility. One can determine from the input conditions which model is necessary, and also easily perform parametric studies to verify assumptions regarding the state of the plasma, or evaluate the importance of non-equilibrium effects. Furthermore, it is also desirable in some circumstances to adjust the level of description according to the *local* conditions. This would maximize efficiency, since higher levels of non-equilibrium imply higher computational costs. The software is currently being configured to allow different non-equilibrium fluid models in different regions; this feature is currently limited to static and pre-determined conditions, but it is theoretically feasible to extend this capability to automatic and time-dependent determination of the non-equilibrium model. This capability implies that boundary conditions between contiguous domains automatically transform the data between lower and higher-levels of

[†] When complex 3D shapes are involved, the Cartesian cells must be "cut" near the boundary, leading to complex tetrahedral shapes. To achieve high accuracy near the boundaries becomes more difficult and efficiency drops

thermodynamic description. Therefore, key methods to be implemented in the library for fluid modeling are for the *contraction* (from higher- to lower level) and *expansion* of information. An example of the former would be the grouping of all excited levels from a C-R model into a Boltzmann distribution and the forcing of equilibrium with the free electrons (from C-R to 2-T model); an example of the latter would be the generation of an additional variable for the electron energy from the single fluid equations (from 1-T to 2-T). Preliminary tests of these procedures have been performed successfully. Of course, the procedures are accurate as long as the conditions permit it, i.e. are such that the higher level of description (e.g. 2-T model) is compatible with local thermodynamic equilibrium (LTE) for the mode being expanded or contracted ($T_e \approx T_h$). This approach yields maximal benefits when the high level of thermodynamic description is very expensive to maintain in near-equilibrium conditions; this would be the case of the C-R model, for example, since it requires a solver for the master equations of collisional and radiative excitation and de-excitation, and mass conservation equations for all levels. Similar benefits could be found for molecular plasmas. Note also that a similar procedure applied to translational modes (i.e. velocity DF) would allow the transition between CFD in dense fluid regions and DSMC in transitional or rarefied regions.

Finally, one should also point out that atomic electronic states are too numerous to be treated independently inside a C-R model; the usual procedure consists of grouping together electronic states which are close together in the energy spectrum.

2.5. Particle Data-Structure

Particles usually have a very simple structure: they must contain a minimum of information, since accurate simulations requiring 10^6-10^7 particles or more would consume an inordinate amount of memory if each particle object contained an excessive amount of data. The minimum information is: (a) a particle type identifier (this can be the species number); (b) particle coordinates (x ,y,z); (c) particle velocity (v_x,v_y,v_z). The rest, i.e. charge, mass, etc, can be obtained from the species information. Additional information may include: (d) particle statistical weight; (e) cell number to which the particle belongs, but also; (f) transformed local coordinates (x, h, z) which are especially useful for monitoring whether a particle crosses into an adjacent cell (and which one) in body-fitted grids, and; (g) time scale – this will be explained further in section 3.1. The statistical weight (each simulated particle is a "pseudo-particle" and represents a large number v of real particles) could be made species-dependent only, in which case it would not need to be included as item (d) in the particle data-structure, but there are several reasons why this may be beneficial: (1) when particles are injected from time-dependent boundary conditions (e.g. electrons from cathodes) a higher accuracy could be obtained by varying the statistical weight in addition to, or instead of, varying the number of pseudo-particles; (2) chemical reactions can be more accurately computed using variable statistical weights; (3) higher statistical accuracy can be achieved when the particle density varies by several orders of magnitude across the computational domain, by duplicating particles that move into low-density regions. These issues are common to DSMC simulations, where the use of variable statistical weights is very frequent [5].

Since some of these quasi-particles may describe gas species with internal structure (electronic, vibrational and rotational states), there may also be some information required to describe these modes. Ideally, to each particle would be associated a unique state; however, to accurately (i.e. with low statistical noise) describe the complete DF, one would require an extraordinarily large number of particles. One must instead attribute a distribution of internal states, e.g. a simulated molecule could have its own rotational temperature, indicating how these modes are populated. During a collisional energy exchange, one can compute and modify the overall internal energy associated with that mode, and assign again a temperature. The disadvantage of this approach is that one potentially neglects correlations between internal and translational modes. For example, if situations occur where most of the excited states (indicated by a quantum number λ) are associated with particles moving at high-velocity in the z-direction, the statistical correlation $\langle \lambda | v_z \rangle$ can be high when both λ and v_z are high, and this effect would not be reproduced if internal energy distributions are stored for each particle. Such situations can occur in special cases (beam interactions) and although molecular plasmas are not currently being considered, it is prudent to design the software with both potential approaches in mind; the data-structure can simply have un-instantiated arrays of levels or distribution parameters, which can be used for whichever model becomes implemented.

This approach also could be extended to the velocity distribution, i.e. the particle data includes also a velocity distribution. This approach is particularly interesting when dealing with the problem of particle coalescence, to be described in the next section.

3. Numerical Model Development

3.1 PIC Model

The PIC model currently in place is a simple electrostatic, non-relativistic model. Particles are generated with variable statistical weight; this is useful in cases where emission is variable, i.e. depends on the plasma and field conditions near the cathode. Only electrons are presently implemented in the PIC model, and electrode boundary conditions are limited to a couple of simplified models. Electrons are "pushed" in three partial steps: first, the velocity is advanced to the time level (n+1/2); second, the particle position is changed using the updated velocity; third, the particle velocity is advanced again to the next time level (n+1). This is the simplest form of a second-order symplectic integration scheme [18]. Note that the velocity must be evaluated at time level (n+1) if one wants to evaluate the current density and inelastic collision rates appropriately at that level. The global time step is of the order of the average time taken by a particle to move by the characteristic length of a cell. There is no strict Courant-like condition for the particles, but that in average the particles will not cross more than one cell per time-step. Exception is made for the particles in cells lying next to a boundary, as explained below.

The "move" phase (second step of the list above) is in itself decomposed into several phases. As noted in section 2.5, each particle data-structure contains a time variable (item (g)), which is the time left to the particle for executing its move. At the start of each iteration, this time-left is increased by the global time step Δt. During the first phase, only the particles that belong to cells near a boundary are being moved (this is a cell-based operation, which obviously requires that the cell contains a list of particles inside the cell). For these, the exact time at which the particle hits the boundary (if at all) is determined, and these particle positions are advanced up to that time; this implies that the time-left is decreased by the corresponding amount. These particles are then flagged and treated by the particle boundary condition procedure. For example, if the boundary is a solid wall, the particle velocity is reflected, and if the boundary is a patch to another grid, the particle is buffered for transmission to another processor. Finally, the particles are moved again by the remaining time-left individual to each particle. This last phase includes a check on the new cell position; if it appears that some particle crosses into a boundary (i.e. was two-layers removed from the boundary, and therefore not considered by the cell-based pre-positioning), its trajectory is re-computed with a smaller elapsed time until it end-up within the computational domain. This procedure prevents the loss of particles from the system; it would affect only a few particles, and only if the global time step is chosen relatively large, and therefore is not a significant computational penalty. Note that such particles would end-up with some time remaining, which will be processed at the *next* time-step. The use of a time-left variable for each particle allows some flexibility in the particle transport, avoiding the processors to spend extraneous time attempting to synchronize all particles. Note also that the check on cell identification from the particle position, at the end of the final "move" phase, is a *Mesh*-based method that is dependent on the grid structure. In the Cartesian case, a few simple divisions provide the answer; in the body-fitted case or unstructured grid case, additional operations may be involved.

A critical issue in the problems of interest here is the high rate of ionization that can be obtained. If each ionization event from a simulated collision of a particle with atoms of the background fluid generates additional particles, the number of simulated particles can grow exponentially, leading to a commensurate growth in computational work. To prevent this, one must devise an algorithm for particle coalescence. Let us first consider the merging of two particles of identical type (and mass m) into a single pseudo-particle of combined statistical weight $\nu_{ab}=\nu_a+\nu_b$, and with a combined velocity $\vec{u}_{ab}=(\nu_a\vec{p}_a+\nu_a\vec{p}_a)/(m\nu_{ab})$. This procedure conserves momentum, but the sum of the kinetic energies of the individual constituents differs from the mean kinetic energy:

$$\begin{aligned} dE = E_{Ka}+E_{Kb}-E_{Kab} &= \nu_a\frac{\vec{p}_a^{\,2}}{2m}+\nu_b\frac{\vec{p}_b^{\,2}}{2m}-\nu_{ab}\frac{m}{2}\vec{u}_{ab}^{\,2} \\ &= \frac{\nu_{ab}}{2m}\frac{\nu_a\nu_b}{(\nu_a+\nu_b)^2}\left[p_a^2+p_b^2-2p_a p_b\cos\theta_{ab}\right] \end{aligned} \tag{1}$$

Note that dE is always positive:

$$\frac{\nu_a \nu_b}{\nu_{ab}} \frac{(p_a - p_b)^2}{2m} \le dE \le \frac{\nu_a \nu_b}{\nu_{ab}} \frac{(p_a + p_b)^2}{2m} \qquad (2)$$

Therefore, to conserve both momentum and energy during the coalescence process, a quasi- internal energy is created, which contains the randomized components of the kinetic energies of the initial particles. This is equivalent to describing the pseudo-particle with a distribution of velocities, such that the temperature of an equivalent Maxwellian distribution is: $dE = \frac{3}{2} kT$. One could add to the description of the particle this "thermal" component, and there are a number of arguments in favor of such a modification. A very sophisticated model based on this approach has been recently described by Hewett [11], and demonstrated high accuracy with fewer number of particles. Besides the reduction in computational cost, there is another potential advantage in having internal velocity DF in the pseudo-particle data structure. When dealing with collisional plasmas, the rates of excitation and ionization are very sensitive to the energy distribution function near the inelastic thresholds, and can be a critical part of the overall plasma dynamics. Using a MCC method with only a few particles may generate an unacceptable level of stochastic noise; if each pseudo-particle also contains its own velocity DF, a smoother and more accurate evaluation of these rates could possibly be obtained. Assuming a Gaussian distribution centered on the mean particle velocity and using a Taylor expansion of the cross-section around that velocity, one can easily evaluate the inelastic rate as function of the pseudo-particle temperature[†]. However, a succession of merging processes would eventually lead to excessive internal DF temperatures, and assigning dynamics based on the mean velocity only (i.e. particle trajectories) leads to inaccuracies. In the absence of the internal DF of each pseudo-particle, it is clear from the discussion above that one cannot combine two particles into a single one while conserving mass, momentum and energy. There have been various schemes [19-20] proposed to minimize the energy error, but the one described below provides an exact conservation scheme.

Let us first consider the more general process of combining an arbitrarily large number of particles, and to consider the energy in the center of mass (CM) frame. Combining N identical particles, we get a total mass $M = \sum_i \nu_i m$, and a total momentum $\vec{P} = \sum_i \vec{p}_i$. The velocity of the center of mass is: $\vec{U}_{CM} = \vec{P} / M$. Denoting by a prime the variables in that CM reference frame, the individual constituents have a momentum: $\vec{p}'_i = \vec{p}_i - m\vec{U}_{CM}$, and contribute a kinetic energy $e'_i = (\vec{p}'_i)^2 / 2m$ to the "thermal" energy. The latter can be expressed as the difference between the total energy contributions of all constituents and the kinetic energy of the center of mass, i.e.:

$$dE = \sum_i \nu_i \frac{\vec{p}_i^{\,2}}{2m} - \frac{\vec{P}^2}{2M} = \sum_i \nu_i \frac{\vec{p}'^{\,2}_i}{2m} \qquad (3)$$

This particle distribution can now be split into two pseudo-particles with half the statistical weight of the entire system. Furthermore, this can be done while transforming the thermal energy into the relative kinetic energy of the two "children". Again, consider the split in the CM frame, into two pseudo-particles of mass $M/2$ with equal and opposite momentum $d\vec{p}'$ (see Figure 4). The total kinetic energy is now:

$$2 \times \frac{(d\vec{p}')^2}{2(M/2)} = dE \qquad (4)$$

Therefore, one can generate two pseudo-particles with half the mass and statistical weight of the center-of-mass, and with equal and opposite momentum in the CM frame, with the magnitude is given by (4), i.e.:

$$|d\vec{p}'| = \sqrt{MdE/2} \qquad (5)$$

and with an orientation chosen at random along the unit solid angle. Note that there is no need for a formal transformation to the CM rest frame; the quantities of interest can be evaluated directly, using (3) for the thermal energy, and with post-fragmentation momentums: $\frac{1}{2}\vec{P} \pm d\vec{p}'$, using (5).

The procedure outlined produces two particles from N particles without creating internal energies, and while exactly conserving mass, momentum and energy. As long as N>2, there is an effective reduction in the number of particles, and while there is no conservative process to coalesce 2 particles into 1, the process of merging 4 into 2 is equivalent. The reduction factor achieved here can take all values (n+2)/2 starting from n=1, and is quite efficient for high values of n.

[†] The procedures for evaluating the inelastic processes are described in section 3.2.

We conducted preliminary tests of this procedure on a simple planar diode case. Without particle merging, the ionization of the gas rapidly leads to exponential growth in the number of simulated electrons. With an aggressive coalescence process in place, we were able to reduce the number of particles from 250,000 to 35,000 with no significant losses in accuracy of the results. Figures 5 and 6 show the potential profiles and particle charge densities obtained in both cases at the same simulated time. Note that initially linear, the potential profile has developed a virtual anode that partially traps electrons and slowly migrates towards the cathode (the ions are immobile in this simulation: the apparent motion is a shift in the ionization front). This also causes the field at the cathode to increase, and the electron emission rate to increase as well. This profile is in agreement with analytical models [3]. Because the potential is obtained as a result of solving Poisson's equation, which filters-out short-wavelength fluctuations, a better test is the comparison of the charge densities themselves. Although Figure 8 shows that the statistical noise is slightly higher, as expected, in the merging case, the results are in excellent agreement. We should point out that currently there is no preferred selection of which particles to be merged, and this can results in artificial thermalization in some cases (e.g. beam-beam interactions); this feature is currently being developed.

3.2 Collisional-Radiative MCC

The intent to implement a Collisional-Radiative model with a PIC model for the high-energy plasma component can lead to some difficulties. Usually, collisions are treated within the framework of the PIC approach with a stochastic method, i.e. Monte-Carlo Collisions (MCC). In this approach, the probability P_c of the modeled pseudo-particle having a collision leading to some event, whether energy exchange, ionization, excitation or other, is compared to a random number R. If the event is "probable" ($R < P_c$), the event is being computed, i.e. in the case of ionization, the pseudo-particle looses the energy required to ionize the target, an additional electron is produced with the same statistical weight as the impacting particle, and the number density of the target is decreased by a corresponding amount. In the case of a C-R model, one must also consider a multiplicity of electronic levels. Consider ionization from any atomic level n by electron impact, as shown in Figure 7. I_n is the ionization potential of that level, and let us assume that this bound-free transition leaves an ejected electron with a negligible kinetic energy. The rate of ionization due to such transitions from level n is:

$$\frac{\partial N_i}{\partial t} = N_n \int_{I_n}^{\infty} de\, s(e) v \cdot f(e) \tag{6}$$

where $v = \sqrt{2e/m}$ (or $v = \sqrt{c^2\left(1 + mc^2/e\right)^{-1}}$ in the relativistic case), e is the kinetic energy of the electron, $f(e)$ is the normalized electron energy distribution function (EEDF). After discretization of the distribution function (see Figure 8):

$$\frac{\partial N_i}{\partial t} \approx N_n \sum_k \Delta e_k\, \bar{f}(e_k) \cdot s(e_k) \cdot \sqrt{2e_k/m} \tag{7}$$

which we can re-write as:

$$\frac{\partial N_i}{\partial t} \approx N_n \sum_k \bar{k}_{i(k)} \cdot \bar{n}_{e(k)}$$

where $$\bar{n}_{e(k)} = \Delta e_k\, \bar{f}(e_k) = \int_{\Delta e_k} de \cdot f(e) \tag{8a}$$

and $$k_{i(k)} = s(e_k) \cdot \sqrt{2e_k/m} \tag{8b}$$

The rate of change is obtained as the sum of elementary processes, such that electrons from kth-bin are moved to another energy bin m such that:

$$e_k - e_m = I_n \tag{9}$$

For each *elementary* process, we have:

$$\frac{\partial N_i}{\partial t} = -\frac{\partial N_n}{\partial t} = +k_{i(k)} \cdot \bar{n}_{e(k)} \cdot N_n \tag{10a}$$

$$\frac{\partial \bar{n}_{e(k)}}{\partial t} = -k_{i(k)} \cdot \bar{n}_{e(k)} \cdot N_n \tag{10b}$$

$$\frac{\partial \bar{n}_{e(m)}}{\partial t} = +k_{i(k)} \cdot \bar{n}_{e(k)} \cdot N_n \tag{10c}$$

$$\frac{\partial \bar{n}_{e(0)}}{\partial t} = +k_{i(k)} \cdot \bar{n}_{e(k)} \cdot N_n \tag{10d}$$

The last equation describes the production of electrons with zero kinetic energy (the ejected atomic electron). The interesting aspect of eqs (10) is that one can construct a complete system of kinetic equations and solve implicitly, i.e. when expanding the RHS to first order in $d\bar{n}, dN$:

$$\begin{bmatrix} 1+\sum n_k dt & -\sum n_k dt & \ldots & \\ \cdots & 1+\sum n'_k dt & & \\ \vdots & & \ddots & \\ & & & 1+\sum n'''_k dt \end{bmatrix} \otimes \begin{bmatrix} dN_i \\ dN_n \\ d\bar{n}_{e(k)} \\ \vdots \end{bmatrix} = \begin{bmatrix} rhs_i \\ rhs_n \\ rhs_{e(k)} \\ \vdots \end{bmatrix} \tag{11}$$

One can then use time scales larger than the characteristic time scale of ionization/recombination, and by extension, excitation/de-excitation for a full C-R model. However, in the PIC approach the EEDF is approximated by a summation of Dirac delta-functions, one for each pseudo-particle, i.e.:

$$n_e = \int_0^\infty d\varepsilon\, f(\varepsilon) = \int_0^\infty d\vec{v}\, g(\vec{v}) = \frac{1}{V}\sum_p \nu_p \delta(\vec{x}-\vec{x}_p)\cdot\delta(\vec{v}-\vec{v}_p) \tag{12}$$

where ν_p is the statistical weight of the particle. Consider now the standard MCC approach. Suppose that one is considering the ionization from an excited level which has a low population density ($N_n \ll N_o$). The rate of ionization due to a given pseudo-particle is given by:

$$\frac{\partial N_i}{\partial t} = N_n \frac{\nu_p}{V} \sigma(\varepsilon_p)\cdot\sqrt{2\varepsilon_p/m} = k_{i(p)} N_n \frac{\nu_p}{V} \tag{13}$$

The probability of this event to occur within a time step Δt is

$$P_c = 1. - e^{-\nu_c \Delta t} \approx \nu_c \Delta t = k_{i(p)} N_n \Delta t \tag{14}$$

and is small when the level population is low. Nevertheless, there will be occasional events when ionization from that level does occur. The MCC procedure would then add to the ion density and remove from the density of excited levels. However, the number density of that excited state would change at that time by:

$$dN_n = -\frac{\nu_p}{V} \tag{15}$$

which can easily lead to *negative* population densities when $\nu_p / V \gg N_n$.

One could imagine a modified procedure where conservation properties (mass, energy) are conserved in *average*, by implementing a small change in the population density at each time step, irrespective of whether the collision event is accepted or not. In that case, the elementary change would be:

$$dN_n = -\frac{\nu_p}{V}\left(k_{i(p)} N_n \Delta t\right) \tag{16}$$

Therefore, if the term in parentheses is sufficiently small, the density of excited states can remain physical. However, this is not always true; even if N_n decreases as the level energy increases, the corresponding cross-section of ionization also increases. Therefore, the product is not always very small, and there is still a chance that the gradual change technique, besides the problem with conservation properties, would not be able to prevent negative population densities. What is needed is an implicit C-R method similar to (11), which also works when some of the rates are provided by interaction with pseudo-particles from a PIC model. There are several approaches to this problem, which are currently being investigated.

3.2 Multi-Temperature Model

The simplest models for the convection of the plasma fluid variables are finite-volume based; in this approach, the fluxes are evaluated at the cell interfaces and Gauss law is used to compute the changes in the cell-averaged variables. This so-called conservative form can be written as:

$$\frac{\partial \mathbf{Q}}{\partial t} + \vec{\nabla}\cdot\vec{\mathbf{F}} = \dot{W} \qquad (17)$$

with $\mathbf{Q}$ the array of conserved variables, $\mathbf{F}$ the array of convective fluxes and $\dot{W}$ the array of local source and exchange terms. This approach is well-suited for the arbitrary grid topology described in section <>; although structured solvers are currently being implemented, we envision at a later stage to supplement the methods used to compute the fluxes by others designed for unstructured grids. Presently, the fluid models being implemented are single- and multi-temperature versions of a second-order Total Variation Diminishing (TVD) scheme. For convection processes only (i.e. the Euler equations) and ignoring the source terms due to collisional exchanges of mass (chemistry) and energy (relaxation), we have:

$$\mathbf{Q}=\begin{bmatrix}\rho_s\\ \rho u^{\alpha}\\ \vdots\\ E\\ \tilde{E}_e\end{bmatrix} \qquad \mathbf{F}^{\beta}=\begin{bmatrix}\rho_s u^{\alpha}\\ \overline{\overline{P}}_{\alpha\beta}\\ \vdots\\ u^{\beta}(E+P)\\ u^{\beta}\tilde{E}_e\end{bmatrix} \qquad \dot{W}=\begin{bmatrix}0\\ 0\\ \vdots\\ 0\\ -P_e\vec{\nabla}\cdot\vec{u}\end{bmatrix} \qquad (18)$$

The total energy E of the plasma includes a purely thermal component $\tilde{E}$ and the kinetic energy. $E=\tilde{E}+\frac{1}{2}\rho\vec{u}^2$. The pressure tensor can be defined as the sum of two contributions:

$$\overline{\overline{P}}_{\alpha\beta} = \underbrace{P\delta_{\alpha\beta}}_{static} + \underbrace{\rho u_{\alpha} u_{\beta}}_{dynamic} \qquad (19)$$

For a two-temperature plasma [21], there are two contributions to the pressure, and to the plasma internal energy, i.e. $P=P_h+P_e$, with $(\gamma_e \equiv 5/3)$:

$$P_h=(\gamma_h-1)\tilde{E}_h\ ;\quad P_e=(\gamma_e-1)\tilde{E}_e \qquad (20)$$

This implies that a jump in pressure across a cell interface can be expressed by:

$$\Delta P=(\gamma_h-1)(\Delta E-\Delta E_K-\Delta\tilde{E}_e)+(\gamma_e-1)\Delta\tilde{E}_e \qquad (21)$$

where:
$$\Delta E_K=\vec{u}\cdot\Delta(\rho\vec{u})-\frac{\vec{u}^2}{2}\Delta\rho=\rho\vec{u}\cdot\Delta\vec{u}+\frac{\vec{u}^2}{2}\Delta\rho \qquad (22)$$

The linearization $\Delta F=\mathbf{A}\circ\Delta Q$ forms the basis of the numerical schemes based on approximate Riemann solvers [22] (so-called Godunov schemes [23]), such as the TVD scheme [24]. Notice that equation (18) contains a source term on the Right Hand Side (RHS) for the electron energy equation. This term is a consequence of the one-fluid approximation, and is responsible for the adiabaticity of the electron gas component. Notably, since the electron gas is always subsonic, there is no shock heating of the electrons across a viscous shock, where there can be a considerable jump in ion temperature [25]. The complete equation for the electron energy, combined with the conservation of mass for the electrons, leads to:

$$T_e^{-1}(D_tT_e)=(2/3)n_e^{-1}(D_tn_e)\ , \qquad (23)$$

where $D_t\equiv\partial_t+\vec{u}\cdot\vec{\nabla}$ is the covariant derivative. Therefore the electron heating (cooling) obtained as a result of compression (expansion) is *always adiabatic*: $\log(T_e)=(\gamma_e-1)\log(n_e)$. The single-fluid formulation has an important problem: the $P_e\vec{\nabla}\vec{u}$ term on the RHS is not an exact flux differential and therefore cannot handle discontinuities. A modified formulation using an entropy-like variable [26] leads to a simpler and more accurate treatment. We define[†]:

$$S_e=\frac{p_e}{\rho^{\gamma_e-1}}\quad ;\quad \hat{s}_e=\frac{p_e}{\rho^{\gamma_e}} \qquad (24)$$

The convection of this quantity yields:

[†] ρ_e was used in the original formulation [1], in which case $\hat{s}_e$ was the true electron entropy, but the advantage here is that the variables in (1) remain definite when the ionization fraction goes to zero ($\rho_e\to 0$, but $\rho\neq 0$).

$$\partial_t\left[r\,\frac{p_e}{r^{?_e}}\right]+\overset{\rho}{\nabla}\left[r\overset{\rho}{u}\,\frac{p_e}{r^{g_e}}\right]=(g_e-1)\,r^{1-g_e}\left[\partial_t\tilde{E}_e+\overset{\rho}{\nabla}(\overset{\rho}{u}\tilde{E}_e)\right]+(1-g_e)\frac{p_e}{r^{g_e}}\left[-r\,\overset{\rho}{\nabla}\cdot\overset{\rho}{u}\right] \tag{25}$$

Using the equation for the electron thermal energy $\partial_t\tilde{E}_e+\overset{\mu}{\nabla}\cdot\left(\overset{\rho}{u}\tilde{E}_e\right)=-p_e\left(\overset{\mu}{\nabla}\cdot\overset{\rho}{u}\right)$, we can easily verify that the modified electron entropy satisfies a simple conservation equation, i.e.:

$$\partial_t S_e+\overset{\mu}{\nabla}(\overset{\rho}{u}S_e)=0 \tag{26}$$

The TVD scheme, modified to include this conservation equation, is completely described for the adiabatic speed of sound in the two-temperature (ion-acoustic) plasma:

$$a^2=g\,\frac{p_h}{r}+g_e\,\frac{p_e}{r} \tag{27}$$

The use of the entropy variable may be convenient for the convective processes, but not for electron heat conduction and inelastic energy exchanges. The latter make it more appropriate to retain the electron thermal energy as the conserved variable. However, the operator-splitting approach allows us to compute the convective processes differently. Thus, the TVD scheme can be used as described above, using the conservation equation for S_e, and yielding a change in the modified entropy variable:

$$S_e^{(n+1)}=S_e^{(n)}+dS_e=S_e^{(n)}-\frac{\Delta t}{\Delta V}\left[\sum_{surfaces}F\right]_{TVD} \tag{28}$$

From which a corresponding change in electron thermal energy due to convection is computed:

$$d\tilde{E}_e=\left[r^{(n)}+dr\right]^{g_e-1}\frac{S_e^{(n)}+dS_e}{g_e-1}-\left[r^{(n)}\right]^{g_e-1}\frac{S_e^{(n)}}{g_e-1} \tag{29}$$

where dr, dS_e are the results of *convection only*, and are computed by the TVD scheme. This overall approach is more accurate and uses less operations than the previous scheme based on both convective and advective terms; the relative errors for the same cases of shock and rarefaction waves are now of the order of 10^{-3} for a shock and 3×10^{-5} across a rarefaction. Figure 9 shows an example of a test of the multi-fluid model, including relaxation effects and precursor heat conduction.

3.3 Multi-Fluid Model

As an alternative to a PIC model for the energetic electrons, one can envision separating the electrons from the rest of the plasma while keeping a continuum model. This is the basis of the multi-fluid model; it assumes that the electrons are sufficiently collisional within themselves to maintain a DF that is close to a Maxwellian, drifting at a mean velocity that is governed by both inertial and electrostatic effects. The main advantage is that computations can be much faster than with a PIC model, while still being able to account for space-charge effects. The disadvantage is that the time scale is governed by a combination of the mean electron velocity and the speed of sound in the electron gas. Therefore, for this approach to be useful, an implicit fluid solver must be devised for the electrons, coupled to the electric field. A first attempt at devising such a scheme has been made. Described in more details in Appendix A, two versions of the scheme have been designed; one which entails the inversion of 7x7 matrices, and is linear-order in the time step dt; a second version that uses 4x4 matrices, but is a second-order expansion, i.e. includes terms proportional to dt^2. Both algorithms, i.e. the 7x7 implementation given by (A20) and the second-order expansion algorithm described in (A22) were implemented and tested. The test case consisted of a simple one-dimensional plasma with reflecting boundaries ("plasma in a box"); the initial state was a fully ionized plasma without recombinations, and with all the electrons moved to the left half of the box. This generated a large internal electric field which accelerated the electrons towards the right side. Without dissipation, the electron motion leads to an oscillatory behavior, as the electrons accumulate on the right side, create a restoring field that accelerates back the other way, and so on. This test case is similar to the one studied in [Shumlak], but with a much more severe initial condition; the initial state in [Shumlak] was a *slight* perturbation of the space charge, in order to have linear superposition of the natural (Langmuir) plasma oscillations with the background fluid at rest. In our case, the plasma oscillations have a much larger amplitude, and non-linear effects (steepening of the profiles) can be detected; nevertheless, it will be seen that one can still obtain an excellent agreement with theory. This test case is also a much more severe test of the stability of the algorithms. Indeed, it was found that the 7x7 algorithm (A20) would collapse when the time step was too large compared to the plasma frequency or the Courant condition $(|u|+c)dt < \Delta x$. The second-order expansion scheme provided greater stability, but damped the oscillations when the time step was such that $(|u|+c)dt > \Delta x$, $w_{pe}dt >> 1$. This is to be expected, since for large time steps the algorithms cease to be time-accurate. It is not yet exactly clear why this damping occurs (albeit at a slow rate) for time steps just below the Courant condition. Further investigations are planned for the near-future.

3.4 MHD

An ideal MHD solver is also in the preliminary stages of being implemented. Based on previous development work, the solver is shock-capturing and has been successfully tested on the standard Brio & Wu problem, both with aligned and misaligned (rotated problem) grids. The scheme is based on a TVD scheme with the MHD characteristic waves, and uses flux-constrained transport concepts [27] to maintain the divergence of the B-field down to machine precision levels. The method works well on body-fitted grids, and is being extended to include multi-temperature and resistive effects. One of the key problems, for example, is to have the scheme valid in the limit of either small or large conductivity. The ideal MHD equations are then obtained by transforming the expressions of the Lorentz force and Joule heating into modified fluxes for conservation laws of combined plasma and field quantities. Resistive effects appear in the dynamics of the field and total energy, i.e.:

$$\frac{\partial B^a}{\partial t} + \nabla_b \left[u^b B^a - u^a B^b\right] = \nabla_b \left[\frac{1}{sm_0}(\nabla^b B^a - \nabla^a B^b)\right] \quad (30a)$$

$$\frac{\partial}{\partial t}\left[E_P + E_M\right] + \nabla_b \left[u^b (E_P + E_M) + u^a (\overline{\overline{P}}^{ab} + T_M^{ab})\right] = \nabla_a \left[\frac{1}{sm_o}\nabla_b T_M^{ab}\right] \quad (30b)$$

where $E_M = \vec{B}^2 / 2m_o$ is the magnetic energy, the pressure tensor $\overline{\overline{P}}_{ab}$ is defined by (19), and

$$\boldsymbol{T}_{\boldsymbol{M}}^{ab} = \frac{\vec{B}^2}{2m_o} d_{ab} - \frac{B_a B_b}{m_o} \quad (31)$$

is the Maxwell tensor. When the conductivity is small, the magnetic diffusion and associated resistive heating (right-hand-side terms in 30) become very large. These must be computed with an implicit method in order to prevent arbitrarily small time-step limitations. A test of this magnetic diffusion scheme is shown in Figure 10. This test consists of a high pressure core region where the field is also inverted. The field discontinuity is being diffused away, and as the magnetic field decays, its energy is converted into plasma heating (here the convective terms are not being computed). The test compared a time-accurate explicit scheme with the implicit version at various time steps; the results show that despite the large time steps the implicit scheme is very accurate, even at the initial discontinuity where the rate of heating is the largest. Comparisons between the loosely-coupled approach ($\vec{j} \times \vec{B}, \vec{j} \cdot \vec{E}$) and the full MHD solver (eqs. 30) were also successfully performed. This MHD solver will be upgraded to include approaches for both low and high-beta plasmas, and incorporated into the general hybrid model.

4. Future Work and Conclusions

Other solvers, not described here, have been or are being implemented (for example, a GMRES method for Poisson’s equation), and discussions of the Equation-Of-State (EOS) and thermo-chemical database have been set aside. More importantly, issues of multiple time-scales have not been described; these are readily apparent when considering the dynamics of both the bulk plasma and the energetic electrons. The latter evolve on much faster time scales than the ions and neutrals, which can be considered as “frozen” during the time scale dt of transport of the electrons in the PIC model. It is not desirable to completely eliminate the PIC approach in favor, for example, of a ballistic model – where electron trajectories are computed one by one in a given field configuration – since this model would not accurately account for fast transient effects (e.g. beam instabilities). However, it is necessary to being able to sub-cycle the PIC model, such that other components are updated on a time-scale Δt that is governed by their own transport, or by the coupling with the fast (PIC) component. For example, while the PIC model is being sub-cycled, the rates of ionization and excitation must be integrated during that time, until the projected changes in population levels become significant. This is not, however, a trivial matter; during the ionization process, new free electrons can themselves become accelerated by the field and become part of the PIC component. Thus, during time scales that can be much smaller than characteristic times of heavy particle transport or relative changes in composition, the inelastic coupling can affect the time-dependent dynamics of the PIC component, and must be considered. The impact on overall scheduling and parallelization strategies must still be evaluated. Therefore, while small time steps can be used for testing and validation in the near future, an effective strategy for dealing with the multiple time scales will be the next major development task.

In conclusion, the development of a general-purpose hybrid model is well under-way, and is rapidly building up capabilities. The strong Object-Oriented approach is absolutely necessary to the success of this project, and the use of a Java-based top-level architecture provides an elegant solution to the problems of portability and accessibility while maintaining a minimum level of performance; the approach also lends itself to the modern paradigm of grid computing, allowing remote steering and flexible access to computing services. Several numerical issues specific to the cases of interest, namely highly energetic particles inside a dense, collisional bulk plasma, have been discussed. New models and approaches to grid, particle and fluid data-structures, particle transport, and kinetics of particle-fluid interaction are being developed. A wide variety of fluid models is also being implemented and tested. The resulting capability, along with upcoming implementations of a relativistic, fully electro-magnetic PIC and Maxwell solvers, is projected to be very useful in the study of the fundamental physics of new classes of plasmas discharges.

5. References

[1] P. V. Akimov et al., *Phys. Plasmas* **8** (2001), 3788.
[2] P.V. Akimov and H. Schamel, *J. App. Phys.* **92** (2002), 1690.
[3] A. V. Kozyrev et al., *J. Appl. Phys*. **74** (1993), 5366 (1993)
[4] C. K. Birdsall, *IEEE Trans. Plasma Sci*. **19** (1991), 65.
[5] G. A. Bird, *Molecular Gas Dynamics and the Direct Simulation of Gas Flows*, Oxford Univ. Press, 1994.
[6] J.-P. Boeuf, L. C. Pitchford, *IEEE Trans. Plasma Sci.* **19** (1991), 286.
[7] K. Nanbu, *IEEE Trans. Plasma Sci.* **28** (2000), 971; V. Serikov, *IEEE Trans. Plasma Sci*. **27** (1999), 1389.
[8] M. E. Jones, D.S. Lemons, R.J. Mason, V.A. Thomas, D. Winske, *J. Comp. Phys*. **123** (1996), 169.
[9] D. J. Larson, *J. Comp. Phys.* **188** (2003), 123.
[10] W. M. Manheimer, M. Lampe and G. Joyce, *J. Comp. Phys.* **138** (1997), 563.
[11] D. W. Hewett,. *J. Comp. Phys*. **189** (2003), 390.
[12] J. Hauser et al., "A Pure Java Parallel Flow Solver", AIAA paper 99-0549.
[13] Java used for scientific computing 2: Japanese PIC
[14] Proceedings of the ACM Workshop on Java for High Performance Network Computing, Stanford University, Palo-Alto, CA, Feb. 28-Mar-1, 1998.
[15] http://math.nist.gov/scimark; J. M. Moreira et al., IBM Research report RC 21166; J. M. Moreira et al., IBM Research report RC 21255; J.M. Moreira et al., *IBM Systems Journal* **39** (2000).
[16] http://java.sun.com/products/jdk/rmi/; http://www.ipd.uka.de/JavaParty/
[17] http://www.ssec.wisc.edu/~billh/visad.html
[18] H. Yoshida, *Phys. Lett.* **A150** (1990), 262.
[19] F. Assous, T. Pougeard-Dulimbert, J. Segre, *J. Comp. Phys*. **187** (2003), 550.
[20] T. Bagdonat and U. Motschmann, *J. Comp. Phys.* **183** (2002), 470.
[21] J.-L. Cambier, "Numerical Methods for TVD Transport and Coupled Relaxing Processes in Gases and Plasmas", AIAA paper 90-1464.
[22] P. Roe, "Characteristic-based Schemes for the Euler Equations", in *Ann Rev. Fluid Mech*. **18**, 337 (1986).
[23] S. Godunov, *Math. Sb*. **47**, 271 (1959).
[24] A. Harten, *J. Comp. Physics* **49**, 357 (1983).
[25] Y. Zeldovich and Y. Raizer, *Physics of Shock Waves and High Temperature Hydrodynamic Phenomena*, vol. II, Academic Press, New-York, 1967.
[26] H. S. Reksoprodjo and R. Agarwal, "A Kinetic Scheme for Numerical Solution of Ideal Magnetohydrodynamics Equations with a Bi-Temperature Model", AIAA 2000-0448.

[27] C. Soria, F. Pontiga and A. Casellanos, *J. Comp. Phys.* **171** (2001), 47.
[28] G. Tóth, *J. Comp. Phys.* **161** (2000), 605.

Acknowledgements
This work was supported by AFOSR under a laboratory Research Task (02PR05COR), Program Manager: Dr. Fariba Fahroo (program initiated by Maj. W. Hilbun).

Appendix A: Electron-Fluid Model

Consider the conservation equations for density and momentum only in one -direction:

$$\partial_t n + \partial_x (nu) = 0 \tag{A-1}$$

$$\partial_t (nu) + \partial_x (nu^2 + p_e/m_e) = 0 \tag{A-2}$$

define an eigenvalue: $c_e^2 = kT_e / m_e$ which is the speed of sound in an isothermal gas. Then (A-1) and (A-2) can be written as $\partial_t Q + \partial_x F = 0$, where Q is the vector of conserved variables and F is the flux in conservative form:

$$\partial_t \begin{pmatrix} n \\ nu \end{pmatrix} + \partial_x \begin{pmatrix} nu \\ n(u^2+c_e^2) \end{pmatrix} = 0 \tag{A-3}$$

We define the transformation matrices:

$$X = \begin{bmatrix} -\dfrac{u-c}{2c} & +\dfrac{1}{2c} \\ +\dfrac{u+c}{2c} & -\dfrac{1}{2c} \end{bmatrix} \tag{A-4a}$$

$$X^{-1} = \begin{bmatrix} 1 & 1 \\ u+c & u-c \end{bmatrix} \tag{A-4b}$$

and the diagonal eigenvalue matrix:

$$\Lambda = \begin{bmatrix} u+c & 0 \\ 0 & u-c \end{bmatrix} \tag{A-4c}$$

The Jacobian of the system (A-3) is obtained as: $A = X^{-1} \cdot \Lambda \cdot X$. The characteristic jumps are:

$$a = X \cdot \Delta Q = \begin{pmatrix} -\left(\dfrac{u-c}{2c}\right)\Delta n + \dfrac{\Delta(nu)}{2c} \\ +\left(\dfrac{u+c}{2c}\right)\Delta n - \dfrac{\Delta(nu)}{2c} \end{pmatrix} \tag{A-5}$$

In order to construct a Godunov scheme that uses flux-limiters on the characteristic variables, it is necessary to verify that: (a) $\Delta Q = X^{-1} \cdot a$; and (b) $\Delta F = X^{-1} \cdot \Lambda \cdot a$. The first relation is trivial to verify. For the second, we first note that:

$$\Lambda \cdot a = \begin{pmatrix} -\dfrac{u^2-c^2}{2c}\Delta n + \dfrac{u+c}{2c}\Delta(nu) \\ +\dfrac{u^2-c^2}{2c}\Delta n - \dfrac{u-c}{2c}\Delta(nu) \end{pmatrix} \tag{A-6}$$

Then, after some algebra:

$$X^{-1} \cdot \Lambda \cdot a = \begin{pmatrix} \Delta(nu) \\ -(u^2 - c^2)\Delta n + 2u\Delta(nu) \end{pmatrix} = \begin{pmatrix} \Delta(nu) \\ \Delta(nu^2) + c^2 \Delta n \end{pmatrix} \tag{A-7}$$

The second term is equal to the flux component of (A-3) as long as c^2 is invariant, in which case $c^2 \Delta n \equiv \Delta(nc^2)$. This is true because the system (A-1,2) describes an isothermal gas: the pressure is controlled by the temperature, which is an external variable here: there is no equation of state (EOS) that determines the temperature from only n, u, and there is no conservation equation for the energy or pressure. The system can be generalized to more than one dimension, where (A-2) extends to:

$$\partial_t (nu^a) + \partial_b (nu^a u^b + d^{ab}\, p_e / m_e) = 0 \tag{A-8}$$

The system (A-3) becomes:

$$\partial_t \begin{pmatrix} n \\ nu_x \\ nu_y \\ nu_z \end{pmatrix} + \partial_x \begin{pmatrix} nu_x \\ n(u_x^2 + c^2) \\ nu_x u_y \\ nu_x u_z \end{pmatrix} + \partial_y \begin{pmatrix} nu_y \\ nu_x u_y \\ n(u_y^2 + c^2) \\ nu_y u_z \end{pmatrix} + \partial_z \begin{pmatrix} nu_z \\ nu_x u_z \\ nu_y u_z \\ n(u_z^2 + c^2) \end{pmatrix} = 0 \tag{A-9}$$

The flux along a direction $\hat{n}$ (with $\hat{t}, \hat{s}$ the correspondingly rotated unit vectors normal to $\hat{n}$) can be obtained through the following matrix definitions:

$$X = \begin{bmatrix} -\frac{(u_n - c)}{2c} & \frac{\hat{n}_x}{2c} & \frac{\hat{n}_y}{2c} & \frac{\hat{n}_z}{2c} \\ +\frac{(u_n + c)}{2c} & -\frac{\hat{n}_x}{2c} & -\frac{\hat{n}_y}{2c} & -\frac{\hat{n}_z}{2c} \\ -u_t & \hat{t}_x & \hat{t}_y & \hat{t}_z \\ -u_s & \hat{s}_x & \hat{s}_y & \hat{s}_z \end{bmatrix} \tag{A-10a}$$

$$X^{-1} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ u_x + \hat{n}_x c & u_x - \hat{n}_x c & \hat{t}_x & \hat{s}_x \\ u_y + \hat{n}_y c & u_y - \hat{n}_y c & \hat{t}_y & \hat{s}_y \\ u_z + \hat{n}_z c & u_z - \hat{n}_z c & \hat{t}_z & \hat{s}_z \end{bmatrix} \tag{A-10b}$$

and the eigenvalue matrix is:

$$\Lambda = \begin{bmatrix} u_n + c & 0 & 0 & 0 \\ 0 & u_n + c & 0 & 0 \\ 0 & 0 & u_n & 0 \\ 0 & 0 & 0 & u_n \end{bmatrix} \tag{A-10c}$$

With these matrices, the flux-limited solver can be constructed. An implicit algorithm can also be constructed, such that:

$$\begin{bmatrix} \mathrm{O} & \mathrm{O} & 0 & 0 \\ -A_{i-1}^{+} & 1 + A_i^{+} - A_i^{-} & A_{i+1}^{-} & 0 \\ 0 & -A_{i-1}^{+} & 1 + A_i^{+} - A_i^{-} & A_{i+1}^{-} \\ 0 & 0 & \mathrm{O} & \mathrm{O} \end{bmatrix} \cdot \begin{bmatrix} \\ dQ_{i-1} \\ dQ_i \\ dQ_{i+1} \end{bmatrix} = \begin{bmatrix} \Delta F_{i-2} \\ \Delta F_{i-1} \\ \Delta F_i \\ \Delta F_{i+1} \end{bmatrix} \tag{A-11}$$

where A^{+}, A^{-} are the Jacobians for positive and negative eigenvalues only. They are constructed by $A^{\pm} = X^{-1} \cdot \Lambda^{\pm} \cdot X$, with: $\Lambda^{+} = \{\mathrm{K}, \max(0, l^a), \mathrm{K}\}$, and $\Lambda^{-} = \{\mathrm{K}, \min(0, l^a), \mathrm{K}\}$.

We can now turn our attention to the right-hand sides of the conservation equations, which have been ignored so far. We will consider first the electrostatic approximation, which implies that only an electric field is present, and all magnetic fields can be neglected. The conservation equation (A-1) does not change, but the momentum equation is now:

$$\partial_t (n_e u_e^a) + \partial_b \left(n_e u_e^a u_e^b + n_e \frac{kT_e}{m_e} \right) = -\frac{e}{m_e} E^a + \frac{(R_T^a + R_U^a)}{m_e} \qquad \text{(A-12)}$$

where $\vec{R}_T \approx -0.71 \cdot n_e \vec{\nabla}(kT_e)$ is a thermal force and

$$\vec{R}_U = m_e n_e \cdot \left[n_{en} (\vec{U}_n - \vec{u}_e) + n_{ei} (\vec{U}_i - \vec{u}_e) \right] \qquad \text{(A-13)}$$

is the frictional force and n_{en}, n_{ei} are the collisional frequencies for momentum exchange. The latter can be simplified when the ions and neutrals are both at the same velocity $\vec{U}$; if the plasma is at sufficiently low density for an ion-slip to occur, the approach described below can easily be extended to the more general case. For simplification, we also neglect the thermal force, which can also be easily added later on.

The second term on the left-hand-side is the advection term which was treated in the previous section. The contributions of each term on the RHS can be examined one by one. The easiest is the collisional momentum exchange. One can write this contribution as follows:

$$d(n_e \vec{u}_e^a) = dt \cdot n_{eh} [n_e U^a - (n_e u_e^a)] \qquad \text{(A-14)}$$

Evaluating terms on the RHS at the *advanced* time level will yield:

$$d(n_e u_e^a) = dt \cdot n_{eh} [n_e U^a - (n_e u_e^a)] + (dt \cdot n_{eh} U^a) dn_e - (dt \cdot n_{eh}) d(n_e u_e^a) \qquad \text{(A-15)}$$

The term involving the electric field can be written in a similar form,

$$d(n_e u_e^a) = -dt \frac{e}{m_e} n_e E^a - dt \frac{e}{m_e} n_e \cdot dE^a - dt \frac{e}{m_e} E^a dn_e \qquad \text{(A-16)}$$

and we need only to express the change in the electric field. Using one of Maxwell's equations:

$$e_o \frac{\partial \vec{E}}{\partial t} + \vec{j} = \frac{\vec{\nabla} \times \vec{B}}{m_o} \qquad \text{(A-17)}$$

and neglecting the magnetic field (electrostatic approximation), this leads to:

$$e_o \partial_t E^a = -e \left(Z_i n_i U_i^a - n_e u_e^a \right) \qquad \text{(A-18)}$$

Again, expressing the RHS at the advanced time level yields:

$$dE^a = -\frac{e}{e_o} dt \left(Z_i n_i U_i^a - n_e u_e^a \right) + \frac{e}{e_o} dt\, d(n_e u_e^a) \qquad \text{(A-19)}$$

One can therefore construct a point-implicit solution of the form:

$$\begin{bmatrix} 1 & 0 & 0 \\ +\frac{eE^a dt}{m_e} - dt \cdot n_{eh} U^a & 1 + dt \cdot n_{eh} & +\frac{e n_e dt}{m_e} \\ 0 & -e dt / e_o & 1 \end{bmatrix} \otimes \begin{pmatrix} dn_e \\ d(n_e u_e^a) \\ dE^a \end{pmatrix} = \begin{pmatrix} 0 \\ dt \cdot n_{eh} n_e (U^a - u_e^a) - dt \frac{e n_e}{m_e} E^a \\ -e dt / e_o (Z_i n_i U^a - n_e u_e^a) \end{pmatrix} \qquad \text{(A-20)}$$

Note that this is a point-implicit part due to the Taylor expansion of the RHS. One should also include the advection process, i.e. the matrices defined in (A-11). Without the energy conservation equation, this would lead to a 7x7 matrix system: 3(momentum) + 3(electric field) + 1(density). However, since the evolution of the electric field does not involve neighboring points, it would be simpler to solve for the change in electric field, and include this solution into the evolution of the momentum. Inserting (A-19) into (A16), we obtain:

$$[1 + w_{pe}^2 dt^2] \cdot d(n_e u_e^a) = -(e dt / m_e) n_e E^a + w_{pe}^2 dt^2 \left(Z_i n_i U_i^a - n_e u_e^a \right) - (e dt / m_e) E^a dn_e \qquad \text{(A-21)}$$

and using $J^a = e(Z_i n_i U_i^a - n_e u_e^a)$,(A20) becomes:

$$\begin{bmatrix} 1 & 0 \\ +(e dt/m_e) E^a - dt \cdot n_{eh} U^a & 1 + n_{eh} dt + w_{pe}^2 dt^2 \end{bmatrix} \otimes \begin{pmatrix} dn_e \\ d(n_e u_e^a) \end{pmatrix} = \begin{pmatrix} 0 \\ -(e dt/m_e) n_e E^a + w_{pe}^2 dt^2 \left(J^a / e \right) \end{pmatrix} \qquad \text{(A-22)}$$

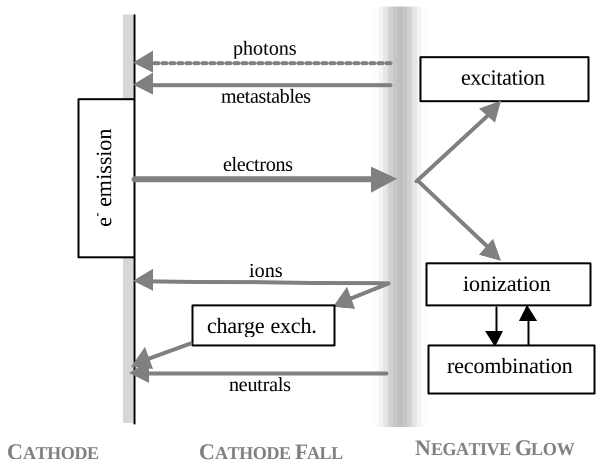

Figure 1: Schematic of physical processes near cathode.

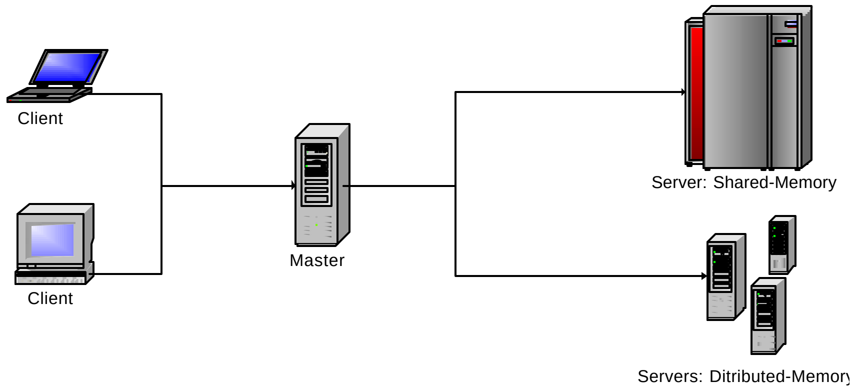

Figure 2: Possible configurations for the Client/Master/Server architecture

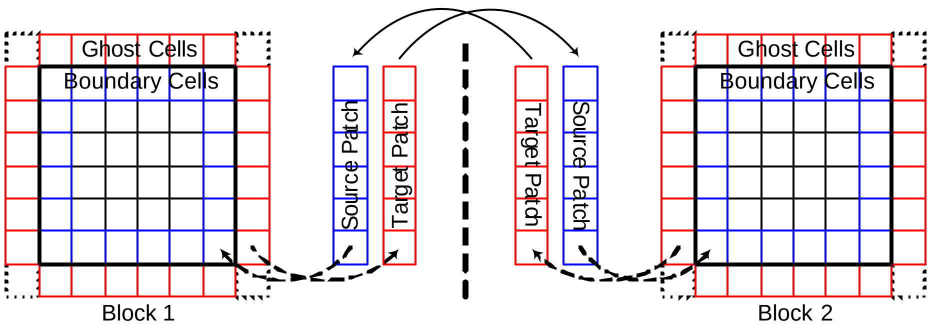

Figure 3: Block communication using source and target patches. The dashed arrows represent referencing while the solid arrows signify remote method calls.

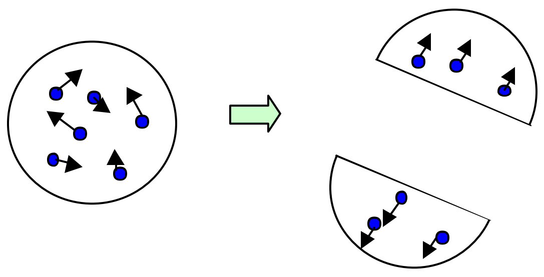

Figure 4: Schematic of fragmentation process in Center of Mass frame after coalescence of many individual pseudo-particles.

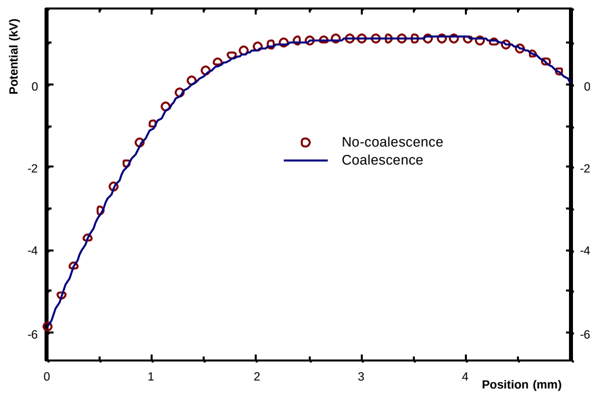

Figure 5: Electric potential in diode 10 ns after start, with and without particle coalescence.

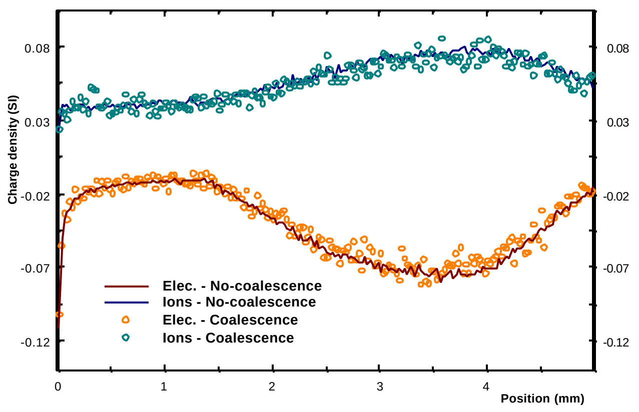

Figure 6: Particle (electron) and fluid (ions) charge densities 10 ns after start of computations, with and without pseudo-particle coalescence. Reduced statistical fluctuations in the former case indicate the approximately 9-times larger number of simulated particles.

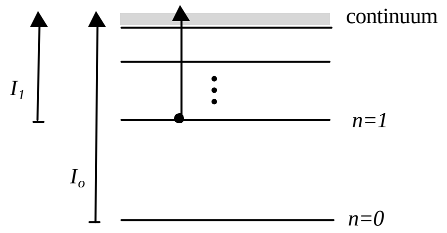

Figure 7: Schematic of ionization scheme in C-R model.

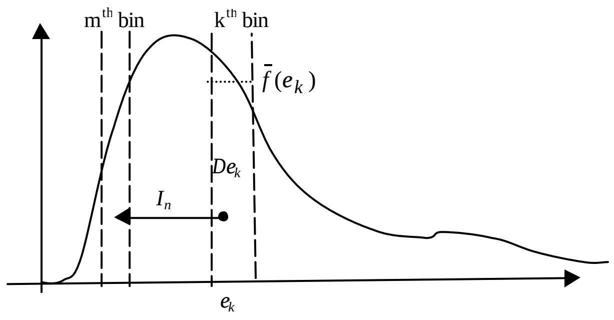

Figure 8: Schematic of discretization of electron distribution function.

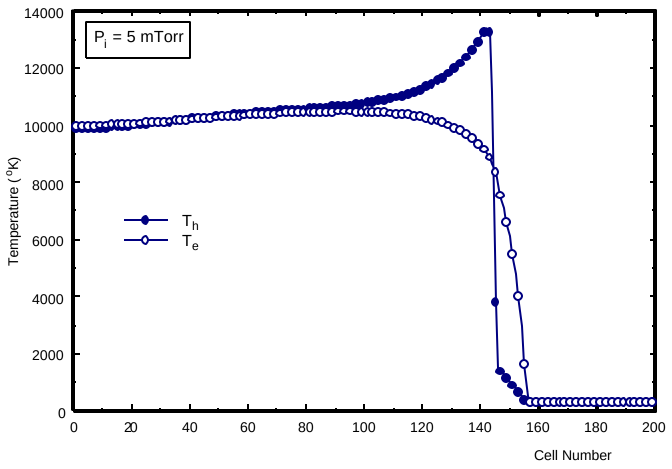

Figure 9: Temperature profiles behind strong shock with relaxation and electron heat conduction.

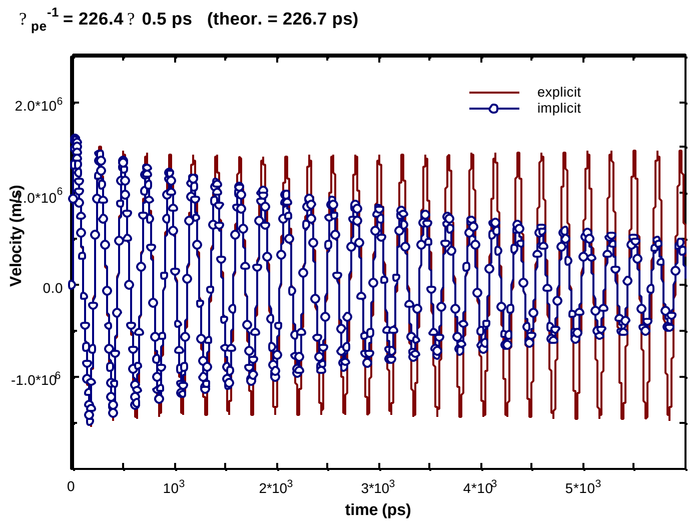

Figure 10: Plasma oscillations (electron velocity at center of 1D box) for explicit and implicit schemes.

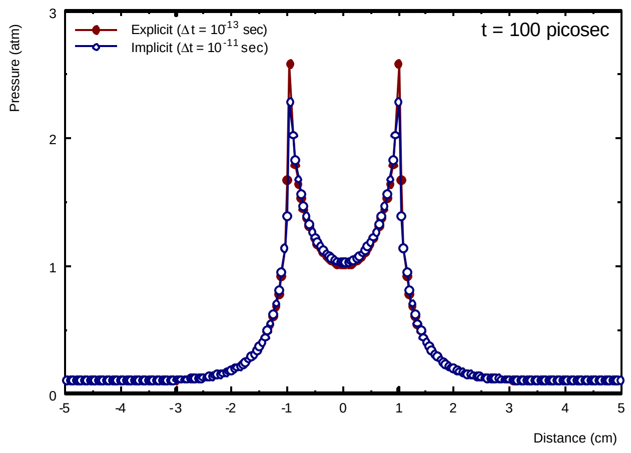

Figure 11: comparison of heating rates (pressure rise) for explicit and implicit magnetic diffusion schemes.